Claude Code Best Practices Guide

As of October 12, 2025 (Version 2.0.14)

Quick Start & Essential Commands

Starting Claude Code

```
# Basic start
claude

# Continue previous conversation
claude --continue
claude --resume

# Start with specific model
claude --model opus
claude --model sonnet

# Add additional working directories
claude --add-dir /path/to/project

# Debug mode for troubleshooting
claude --debug
```

Essential Keyboard Shortcuts

Shortcut	Action	Version Added
Tab	Toggle thinking mode (sticky)	v2.0.0
Tab	Auto-complete file/folder names	v1.0.75
Shift+Tab	Toggle auto-accept for file edits	v1.0.47
Ctrl+R	Search command history (like bash)	v1.0.117 / v2.0.0
Ctrl+O	Toggle transcript view	v1.0.113
Ctrl+B	Run bash command in background	v1.0.71
Ctrl+G	Edit prompt in system editor	v2.0.10
Ctrl+Z	Suspend Claude Code (resume with `fg`)	v1.0.44
Ctrl+_	Undo prompt input	v1.0.45
Esc	Interrupt Claude (in auto-accept mode)	v1.0.70
Enter	Queue messages while Claude works	v1.0.75
Alt+M	Switch modes (Windows)	v1.0.54

Vim Mode Shortcuts (when enabled via `/vim` or `/config`)

```
# Movement
h, j, k, l      # Basic movement
w, b            # Word movement
0, $            # Line start/end
f/F, t/T        # Find character
c               # Change
```

```
# Editing
u              # Undo
```

File & Project Management

@-Mentioning Files and Resources

```
# Mention specific files
@path/to/file.py

# Mention files with spaces (v1.0.83)
@"path with spaces/file.txt"

# Mention directories
@/home/user/project/

# Mention hidden files (v1.0.64)
@.env
@.gitignore

# Mention Claude configuration files (v1.0.84)
@~/.claude/agents/
@~/.claude/output-styles/
@~/.claude/commands/

# Mention MCP resources (v1.0.27)
@mcp-resource-name

# Mention custom agents (v1.0.62)
@my-custom-agent
```

Working with Multiple Directories

```
# Add additional working directories
/add-dir /path/to/another/project
/add-dir ~/Documents/project2

# Command line flag
claude --add-dir /path/to/project
```

File Operations Best Practices

```
# Drag and drop files into chat (IDE) - v2.0.8
# Paste images directly (Ctrl+V or drag) - v1.0.75
# Windows: Alt+V for images - v1.0.93

# Read PDFs directly (v1.0.58)
@document.pdf
```

Conversation Management

Session Commands

```
# View conversation history
/history

# Resume specific conversation
/resume

# Continue last conversation
claude --continue

# Export conversation for sharing (v1.0.44)
/export

# Rewind conversation & undo code changes (v2.0.0)
/rewind

# Clear conversation
/clear
```

Memory Management

```
# Quick add to memory (start message with #)
# This feature helps Claude remember important context
#ProjectAPI uses port 3000 for development

# Edit memory files directly (v1.0.94)
/memory

# View memory contents
/memory show
```

Model Selection & Thinking Modes

Model Selection

```
# Select model interactively
/model

# Use Opus for planning, Sonnet for execution (v1.0.77)
/model opusplan

# Command line model selection
claude --model opus
claude --model sonnet

# Environment variables for default models (v1.0.88)
export ANTHROPIC_DEFAULT_OPUS_MODEL="claude-opus-4-1-20250805"
export ANTHROPIC_DEFAULT_SONNET_MODEL="claude-sonnet-4-20241022"
```

Thinking Modes

```
# Toggle thinking mode
Tab                    # Toggle (sticky across sessions)

# Trigger thinking in prompt
"think about this"     # Auto-triggers thinking
"think harder"         # Deeper thinking
"ultrathink"          # Maximum thinking

# Temporarily disable thinking (v1.0.115)
/t <your prompt>      # Disables thinking for this message

# Disable interleaved thinking
export DISABLE_INTERLEAVED_THINKING=1
```

Tool Permissions & Security

Managing Permissions

```
# View and manage permissions
/permissions

# Configure ask mode for specific tools (v1.0.72)
/permissions
# Then select tools to always ask for confirmation

# Auto-approve specific commands
# Edit settings.json and add to allowedTools:
"Bash(git add:*)"
"Bash(npm install:*)"
"Read(/home/user/project/**)"

# Skip all permissions (use cautiously)
claude --dangerously-skip-permissions
```

Security Best Practices

```
// In ~/.claude/settings.json
{
  "allowedTools": [
    "Bash(git:*)",
    "Bash(npm:*)",
    "Read(/home/user/safe-directory/**)",
    "Write(/tmp/**)"
  ],
  "deniedTools": [
    "Bash(rm -rf:*)",
    "Write(/etc/**)"
  ]
}
```

Custom Extensions & Plugins

Custom Agents (v1.0.60)

```
# View available agents
/agents

# Create custom agent
# Place .md file in ~/.claude/agents/
# Example: ~/.claude/agents/code-reviewer.md

# Mention custom agent
@code-reviewer review this function
```

Output Styles (v1.0.81)

```
# View/select output styles
/output-style

# Built-in styles
- Explanatory  # Educational with detailed explanations
- Learning     # Step-by-step learning approach

# Create custom style
# Place .md file in ~/.claude/output-styles/
```

Custom Slash Commands

```
# Create custom command
# Place .md file in ~/.claude/commands/
# Example: ~/.claude/commands/deploy.md

# Use custom command
/deploy production

# Nested commands (v1.0.45)
# ~/.claude/commands/frontend/component.md
/frontend:component Button
```

Plugin System (v2.0.12)

```
# Install plugins
/plugin install <plugin-name>

# Manage plugins
/plugin enable <plugin-name>
/plugin disable <plugin-name>

# Browse marketplaces
/plugin marketplace
```

```
# Validate plugin structure
/plugin validate <path>
```

Background Tasks & Monitoring

Background Bash Commands (v1.0.71)

```
# Run command in background
Ctrl+B
# Then type your command
npm run dev

# Monitor background processes
/bashes              # List all background shells

# Read output from background process
BashOutput <shell-id>

# Kill background process
KillShell <shell-id>
```

Long-Running Tasks

```
# Start development server in background
Ctrl+B
python manage.py runserver

# Tail logs in background
Ctrl+B
tail -f application.log

# Monitor with filtering
BashOutput <shell-id> --filter "ERROR"
```

IDE Integration

VS Code Extension (v2.0.0)

```
# Features available in VS Code:
- Native extension support
- Drag and drop files/folders
- Image paste support
- Diff visualization
- "Open in Terminal" option

# Disable auto-connection (v1.0.61)
export CLAUDE_CODE_AUTO_CONNECT_IDE=false
```

Debugging & Troubleshooting

Diagnostic Commands

```
# Comprehensive system check
/doctor

# View context usage
/context

# Check system and account status
/status

# View current costs
/cost

# View usage against plan limits (v2.0.0)
/usage

# List current todos (v1.0.94)
/todos
```

Common Fixes

```
# Settings file issues
/doctor  # Validates settings.json

# Force software rendering (GPU issues)
export LIBGL_ALWAYS_SOFTWARE=true

# Skip login shell for faster bash
export CLAUDE_BASH_NO_LOGIN=1

# Fix Anaconda conflicts
export CLAUDE_BASH_MAINTAIN_PROJECT_WORKING_DIR=1

# Proxy configuration
export NO_PROXY="localhost,127.0.0.1"
```

Advanced Configuration

CLAUDE.md Files

```
# Project-level: /project/CLAUDE.md
# Desktop-level: ~/Desktop/CLAUDE.md
# User-level: ~/.claude/CLAUDE.md

# Import other files (v1.0.107)
@path/to/additional/instructions.md
```

Settings.json Configuration

```
// ~/.claude/settings.json
{
  "allowedTools": ["Bash(git:*)", "Read(**/*.py)"],
  "deniedTools": ["Bash(rm -rf:*)"],
  "autocompactThreshold": 0.8,
  "spinnerTipsEnabled": false,
  "disableAllHooks": false,
  "cleanupPeriodDays": 7,
  "forceLoginMethod": "oauth",
  "extraKnownMarketplaces": ["https://my-marketplace.com"]
}
```

Hooks System (v1.0.38)

```
# Hook types available:
- UserPromptSubmit   # Before user message
- PreToolUse         # Before tool execution
- PostToolUse        # After tool execution
- SessionStart       # New session begins
- SessionEnd         # Session ends
- PreCompact         # Before conversation compaction
- Stop               # When Claude stops
- SubagentStop       # When subagent stops

# Configure in settings.json
"hooks": {
  "UserPromptSubmit": {
    "command": "echo 'User said: {input}'",
    "timeout": 5000
  }
}
```

Environment Variables

```
# Model selection
export ANTHROPIC_DEFAULT_OPUS_MODEL="claude-opus-4-1-20250805"
export ANTHROPIC_DEFAULT_SONNET_MODEL="claude-sonnet-4-20241022"

# Timeouts
export BASH_DEFAULT_TIMEOUT_MS=120000
export BASH_MAX_TIMEOUT_MS=600000
export MCP_TIMEOUT=30000

# Shell configuration
export CLAUDE_BASH_NO_LOGIN=1
export CLAUDE_BASH_MAINTAIN_PROJECT_WORKING_DIR=1
export CLAUDE_CODE_SHELL_PREFIX="custom-prefix"

# Debugging
export ANTHROPIC_LOG=debug
export USE_BUILTIN_RIPGREP=0
```

Pro Tips & Best Practices

Performance Optimization

1. **Use thinking mode wisely** - Tab to toggle, great for complex problems
2. **Queue messages** - Press Enter while Claude works to add follow-up tasks
3. **Parallel tool use** - Claude automatically runs independent operations in parallel
4. **Background tasks** - Use Ctrl+B for long-running processes
5. **Auto-compact** - Automatic at 80% context usage

Workflow Optimization

1. **@-mention files** instead of copying content
2. **Use /rewind** to undo problematic changes
3. **Create custom agents** for repetitive specialized tasks
4. **Set up output styles** for consistent formatting
5. **Configure allowed tools** to skip permission prompts

Security Best Practices

1. **Review permissions** with `/permissions`
2. **Use project-level CLAUDE.md** for team instructions
3. **Configure denied tools** in settings.json
4. **Be cautious with `--dangerously-skip-permissions`**
5. **Validate plugins** before installation with `/plugin validate`

Status Line Customization (v1.0.71)

```
# Configure your status line
/statusline

# Information available in status line:
- Session cost
- Token usage indicator
- Current model
- Background processes
- Exceeds 200k tokens warning
```

Web & Search Features

Web Search (v1.0.105)

```
# Claude can search the web automatically
# Just ask questions about current events or recent information

# Web search considers today's date from environment
# Searches are US-based only
```

Web Fetch

```
# Claude automatically fetches URLs you paste
https://example.com

# With specific extraction prompt
"Get the pricing information from https://example.com"
```

Quick Reference Card

Most Used Commands

```
/help              # Get help
/model             # Change model
/permissions       # Manage tool permissions
/context           # View context usage
/doctor            # Diagnose issues
/export            # Export conversation
/rewind            # Undo changes
/agents            # Manage agents
/plugin            # Manage plugins
/todos             # View todo list
```

Quick Actions

```
Tab                # Toggle thinking
Ctrl+R             # Search history
Ctrl+B             # Background command
Ctrl+O             # View transcript
Shift+Tab          # Toggle auto-accept
@file              # Mention file
#note              # Add to memory
/t prompt          # Skip thinking once
```

Latest Features (Version 2.0.14)

- **Plugin System**: Full marketplace support with install/enable/disable
- **Improved Rendering**: Buttery smooth terminal UI
- **MCP Server Toggle**: @-mention to enable/disable
- **Tab Completion**: For shell commands in bash mode
- **Prompt Editing**: Ctrl+G to edit in external editor
- **Reduced Tokens**: System prompt 1.4k tokens smaller
- **Better Permissions**: Inline env var support in bash

Resources

- **Documentation**: https://docs.claude.com/en/docs/claude-code/
- **Plugin Docs**: https://docs.claude.com/en/docs/claude-code/plugins
- **Output Styles**: https://docs.claude.com/en/docs/claude-code/output-styles

- **Slash Commands**: https://docs.claude.com/en/docs/claude-code/slash-commands
- **Hooks System**: https://docs.claude.com/en/docs/claude-code/hooks
- **GitHub Issues**: https://github.com/anthropics/claude-code/issues
- **Plugin Announcement**: https://www.anthropic.com/news/claude-code-plugins

---

This guide is current as of Claude Code v2.0.14 (October 12, 2025)